GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION
SPONSORED PROJECT INITIATION

Date: 8/7/78

Project Title: Provide a Complete State-of-the-Art Analysis of Resource Estimating Methodologies

Project No: E-24-672

Project Director: Dr. H. M. Wadsworth

Sponsor: Battelle Columbus Laboratories; Columbus, OH 43201

Agreement Period: From 6/15/78 Until 9/15/78

Type Agreement: Delivery Order No. 0913 (Time and Materials Subcontract Under Prime DAAG29-76-D-0100)

Amount: $10,645 Ceiling Price

Reports Required: Summary of Results; Reports or SOW Items; Final Report

Sponsor Contact Person (s):

Technical Matters
Contracting Officer's Technical Representative (COTR)
Mr. James M. Irwin
DIR, AIRMICS
313 Calculator Building
Atlanta, GA 30308

Contractual Matters
(thru OCA)
Mr. Jack Franklin
Contracting Officer
Battelle Columbus Laboratories
Durham Operations
3333 Chapel Hill Blvd.
P. O. Box 8796
Durham, NC 27707
Phone (919) 489-2366

Defense Priority Rating: None

Assigned to: Industrial and Systems Engineering (School/Laboratory)

COPIES TO:

- Project Director
- Division Chief (EES)
- School/Laboratory Director
- Dean/Director—EES
- Accounting Office
- Procurement Office
- Security Coordinator (OCA)
- Reports Coordinator (OCA) ✓
- Library, Technical Reports Section
- EES Information Office
- EES Reports & Procedures
- Project File (OCA)
- Project Code (GTRI)
- Other _____

CA-3 (3/76)

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION

Date: 1/17/79

Project Title: ~~Provide~~ a Complete State-of-the-Art Analysis of Resource Estimating Methodologies

Project No: E-24-672

Project Director: Dr. H. M. Wadsworth

Sponsor: Battelle Columbus Laboratories; Columbus, OH 43201

Effective Termination Date: Period of Performance and Reporting Period: 10/15/78

Clearance of Accounting Charges: Same as above

Grant/Contract Closeout Actions Remaining:

- X Final Invoice and Closing Documents
- _ Final Fiscal Report
- X Final Report of Inventions
- _ Govt. Property Inventory & Related Certificate
- _ Classified Material Certificate
- _ Other _____

Assigned to: Industrial & Systems Engi ng (School/Laboratory)

COPIES TO:

Project Director
Division Chief (EES)
School/Laboratory Director
Dean/Director—EES
Accounting Office
Procurement Offic
Security Coor ator (OCA)
Report- ordinator (OCA)
Library, Technical Reports Section
Office of Computing Services
Director, Physical Plant
EES Information Office
Project File (OCA)
Project Code (GTRI)
Other _____

—4 (3/76)

"A Complete State-of-the-Art
Analysis of Resource Estimating Methodologies."

Contract DAAG 29-76-D-0100
Delivery Order 0913

For: Battelle Columbus Laboratories
Columbus, OH 43201

U.S. Army Institute for Research in Management
Information and Computer Science
Atlanta, Georgia 30332

FINAL REPORT

For: Battelle Columbus Laboratories, Columbus, OH 43201

U.S. Army Institute for Research in Management

Information and Computer Science,

Atlanta, Georgia 30332

Contract DAAG 29-76-D-0100, Delivery Order 0913

Title: "A Complete State-of-the-Art Analysis of Resource Estimating Methodologies."

Conducted By: The School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

Harrison M. Wadsworth, Jr.
Principal Investigator

October 1978

The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

A COMPLETE STATE-OF-THE-ART ANALYSIS OF RESOURCE ESTIMATING METHODOLOGIES

INTRODUCTION

There have been many examples which illustrate the problem of life cycle cost estimating for computer software systems. In fact, very few, if any, completed major software development projects have met four typical objectives for successful system development.

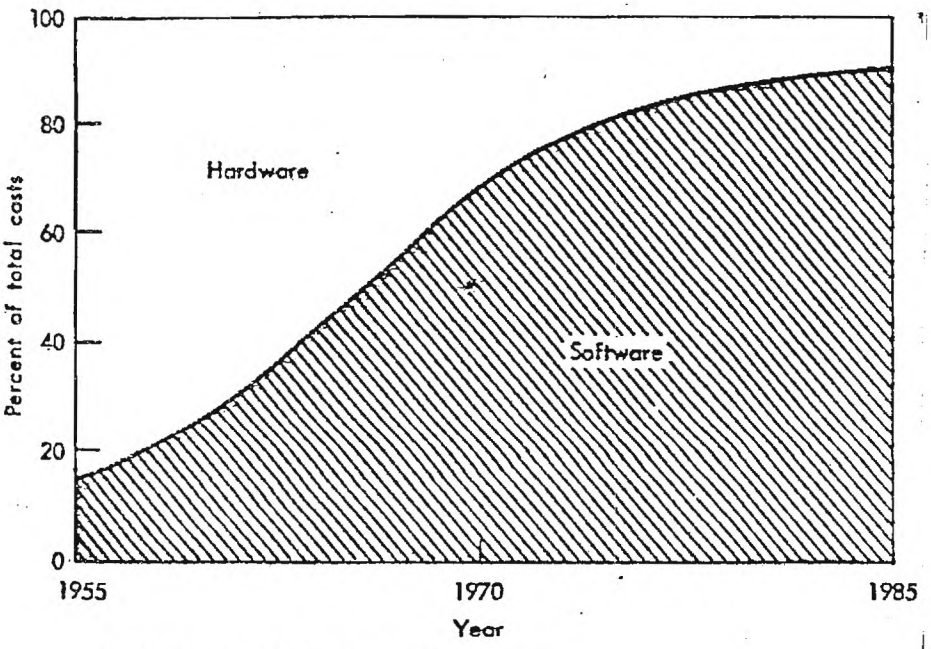
1. Preliminary cost estimates must be accurate within reasonable bounds.
2. Production schedules must be met.
3. Operational requirements and design performance criteria must be met.
4. The final product must be reasonably error free and reliable when installed in an operational environment (21,36).

The objective of producing reliable software is just as important and certainly not at odds with the objective of reducing the high cost of software. There are many reasons for failure to make an accurate preliminary cost estimate. In most cases these estimates are much less than the final cost which suggests they may be underbid in order to obtain a contract to do the work.

Most of the software cost estimating procedures make use of estimates of the program size. This is difficult to estimate as evidenced by many examples in the literature of a final program size of thousands of instructions more than initial estimates.

The importance of the problem of accurate cost estimates is evidenced by the fact that software system development is largely people-oriented work. Computer hardware costs have been decreasing while the cost of software is increasing as are most such costs. As a result, the proportion of

the total cost of a new or revised computer system due to software is rapidly increasing and is now the major part of the cost. This is illustrated by the following figure presented by Boehm (3). For this reason,



Hardware/software cost trends.

if software costs cannot be estimated with reasonable accuracy the cost of a weapon system using such software cannot be estimated in advance. The implications of such a situation are quite obvious.

Research into the problem of accurately forecasting software development cost has not been particularly successful. In fact in some cases it has been disastrous. This has been caused in a number of instances by research sponsors who wished to demonstrate a solution which justified a particular expenditure of research money. For example, in the 1960's the U.S. Air Force spent hundreds of thousands of dollars on this problem. They received a set of equations designed to assist in estimating man hours, etc. for programming and testing computer programs. The research agency dis-

claimed the accuracy of these equations for various reasons. The foremost reason was inadequate data gathered largely by questionaire after the projects had been completed (18).

There are now available a number of differenct procedures which may be used to forecast software development costs. This report will review many of these procedures and attempt to criticize them. The criticisms will include apparent problems or inadequancies found in the procedures. Suggestions for improvement will be made if possible.

Before a reasonably accurate forecasting algorithm can be developed research should be done which will reveal the underlying idiosyncrasies of the software development process. It is because of these idiosyncrasies that well developed hardware cost estimating procedures have not worked. The necessary data for such research is just beginning to be developed (7). The data however should be carefully scrutinized to ascertain if it is really appropriate, and will give the answers needed.

The fact that research into software cost forecasting is needed cannot be refuted. Bratman and Court (5) list five conditions which contribute to the current lack of understanding of the software development process and the inability to forecast development time and resources needed:

1. Lack of discipline and repeatibility
2. Lack of development visability
3. Changing performance requirements
4. Lack of design verification tools
5. Lack of software reuseability

ESTIMATING METHODS

Estimating methods can be divided into categories in various manners. Nelson considers the following categories which depend on the input needed to develop the estimate (24).

1. Specific Analogy. These estimating procedures use known costs of similar projects to determine estimates. They have the advantage of being easily understood and believed, but the disadvantage that no project may be available which is an adequate analogy of the new project.
2. Unit Price. Unit price procedures involve the multiplying of a previously determined cost per unit for a given portion of a system by the number of units to be delivered in the new system. This may involve, for example, cost per instruction or cost per module. Again the advantage is simplicity, but the disadvantages to this method are that it may not take into account the fact that some units may be more costly than others. Even if the cost averages out over the different units, there is still a great deal of difficulty in estimating the number of units.
3. Percent of Other Item. These procedures set the price of one part of the new system as a percentage of another part which may be known or estimated with greater precision. The same advantages and disadvantages which apply to the method of specific analogy apply to these procedures.
4. Expert Opinion. This estimating procedure is not really different in that the experts who make the estimate are using one or more of the previously mentioned methods. It is listed as it has been promoted by some authors in the form of the Delphi Method. This method is simply a procedure which attempts to arrive at a consensus opinion of a group of experts. Farquhar (11) reports on a study run by the Rand Corporation, the originators of the Delphi Method, which indicated that the Delphi consensus opinion was no

better than a simple group judgement. In fact he indicated that neither method was particularly useful for software cost estimating. Dunn (10) has reported on the use of the Delphi method in conjuntion with a risk analysis model which improved on the suitability of the Delphi estimate.

5. Parametric Equations. These procedures use one or more equations that represent the cost of a proposed project or a portion of the project as a function of various characteristics of the system. While such procedures hold a lot of promise, a lack of adequate data has prevented proper validation of the several models that have been presented by various authors. They also may involve the disadvantage noted previously of requiring the estimator be able to count in advance the number of instructions or modules in the system. The ideal procedure of this type would be one which uses as input known or easily determined system characteristics. Current work should be towards developing such estimating procedures.

Many of the currently used parametric estimating procedures seem to suffer from one or more of these basic problems:

1. Lack of hard data for validation purposes.
2. Requirement for input data which is not readily available.
3. Inadequate definitions of factors used in the models.

All of these considerations should be considered by future researchers who may wish to refine current models or develop additional ones.

SPECIFIC ANALOGY PROCEDURES

These procedures are used quite frequently. For example, Lecht (19) describes a method which depends almost entirely on cost data from similar

projects to make estimates of a proposed system. Since this technique uses the past experience of the estimator, it should be restricted to the scope and size of projects previously produced. Persons using it often extrapolate to larger projects assuming a linear relationship. This has proved to be an incorrect assumption in many instances, but the correct relationship has not been determined. Research could be conducted which would develop such relationships for various types of systems but the data required for such research does not appear to be available at the present time.

The specific analogy procedure forms the basis of a formal software cost estimating technique used by the U.S. Army. An army catalog provides detailed cost information for 20 data systems (8). The procedure recommends that an estimator study the systems cataloged and then use adjustments to account for differences between the new systems and the systems cataloged. Procedures for making these adjustments are explained in the handbook.

This method is subjective in nature, but is does tend to reduce the likelihood of overly optimistic schedules and estimates since it is based on actual data for similar systems. If adequate historical data were available, this technique would hold much promise.

The Rome Air Development Center of the Air Force Systems Command is currently attempting to develop the historical data base required for this type of cost estimating. This research has been reported by Nelson and Sukert (26). If this project is successful the problem of extrapolating from the RADC data base to a new system will remain. Hopefully the research conducted by RADC will also consider this problem.

UNIT PRICE PROCEDURES

This procedure is probably the most frequently used technique for

software cost estimating. An expected or average cost for a single computer instruction in a certain language must first be determined. Then the number of such instructions must be estimated. The product of these is then determined and necessary allowances are then added to make the final cost estimate. The relative simplicity and understandability of this technique probably accounts for its frequent use.

The problems associated with this technique are:

1. It many be difficult to determine an appropriate cost per instruction because of a lack of an adequate historical data base. Cost per instructions may depend on the type of system being developed. Available data does not differentiate among these things well enough.
2. It is difficult to estimate the number of instructions in a proposed system. In fact, this may be just as difficult as the prediction of overall software development cost.
3. It is questionable whether an estimate based on the number of instructions is a valid estimate.

These problems are considered by Devenny (9) and will be discussed now in detail. It is difficult to define what is meant by cost and what cost elements are included. Since there is no standard definition, the use of the term will not be consistent. Wolverton (38) suggests that computer time might represent as much as 25% of software development cost while other authors do not even include such a cost. Some cost factors may be in terms of man-hour per instructions while others are in terms of salary dollars. The two procedures would not be comparable.

Another difficulty is that the exact beginning and end of the software development process is not well defined. Some cost per instruction

factors include only the design, coding and debugging phases while others include activities before and after these phases. Such activities often include a major portion of the development cost. A standard definition of the term "cost" must be developed. This definition would include the elements of cost included in the term "cost per instruction" as well as a precise definition of the word "instruction." Such a standard must be mutually agreed upon by all software cost estimators before this procedure can be used with any validity.

Unfortunately the word "instruction" does not have a consistent meaning either. Some estimators include only source instructions whereas others include object instructions. Since object instructions are generated by the source instructions one would not use both. The number of instructions needed for a program depends on the type language used. There are differences even among the high order languages such as FORTRAN or COBOL. There are also differences in amount of difficulty needed to write various types of instructions. Nelson (25) and Frederic (13) both report that by classifying instructions into categories they improved the correlation between cost and the number of instructions in a program.

Frederic, in discussing a research project conducted by Tecolote Research, brings out some of the problems of using "cost per instruction" as as estimating parameter. At the start of their research, they had historical data from 387 different software development projects. All but five of these data points had to be abandoned because of a lack of specific information concerning cost elements and types of instructions counted even though all 387 reports had some cost and instruction count information. As a result of this they were forced to develop their estimating methodology using only five data points rather than the full 387. Their results

will be discussed later in this report.

Even if a valid cost per instruction factor is developed, the estimator is still faced with the problem of estimating the number of instructions. Schwartz (35), for example, reports that a development project conducted by UNIVAC for United Air Lines had an initial estimate of 9,000 instructions. The system was finally cancelled when it had grown to 146,000 instructions and the end was still not in sight.

Clearly the ability to estimate the number of instructions in a proposed program depends on the skill of the estimator and on the amount of design detail which he has. If the program has been broken down into small modules, he is more likely to be able to arrive at an accurate estimate.

Even if the above two problems can be overcome, many authors have questioned whether cost per instruction and number of instructions can accurately predict system development cost. The cost per instruction factor does not specifically address many of the factors influencing cost. A study by the System Development Corporation for the Electronic Systems Division of the Air Force Systems Command identified 94 such factors. Number of instructions is but one of these 94 variables (22).

Another problem with this procedure is that the relationship between number of instructions and cost has not been determined. Many estimators assume it is linear and apply the same factor to a program regardless of its size. Some research performed on this subject using regression techniques has determined the relationship to be nonlinear. For example, Farr and Nanus (12), working for SDC developed the following model:

$$\text{Man-Months of Effort} = (\text{constant}) \times (\text{No. of instructions})^{1.5}$$

Frederic (13), working for Tecolote Research, developed the following model:

$$\text{Cost} = 0.079\ (\text{No. of instructions})^{0.84}$$

where cost is in thousands of 1973 dollars.

Malone (20), working for IBM, developed a third model using the number of source statements in some programs developed at IBM. This model is

$$\text{Man-Months} = 0.00007\ (\text{No. of instructions})^{1.1386}$$

The use of these three models would result in significantly different cost estimates so we must conclude that they cannot all be correct. In fact, none of the three authors recommend the use of their model due to problems of limited data, small samples, and low correlation between their estimates and true costs. They will all be discussed in some detail later in this report.

PERCENT OF OTHER ITEM PROCEDURES

This technique uses the cost of one part of a weapon system to estimate the cost of another part. This procedure has worked quite well for hardware cost estimating. For example the cost of spare parts may be a given percentage of the cost of manufacturing the equipment. However applying it to software is quite difficult.

First, there is the problem of insufficient data to provide adequate cost information. Secondly, software comprises anywhere from 2-90% of the total cost of the system. Thus the use of an average cost factor will be difficult. Several authors have suggested that designing, coding and testing a program might be expected to average 40%, 20% and 40% of the total program cost, respectively. When the design has been completed a manager could use these figures to estimate the cost of the remaining portions of the development project. Boehm (3), however, points out that there are significant differences among these percentages for different projects. Therefore any estimates using them must be very approximate at best.

EXPERT OPINION

This technique involves one or more experts, either individually or as a group, using their knowledge and experience to predict the cost of developing a software system. There are two basic procedures to obtain this prediction: engineering cost analysis and the Delphi method.

For an engineering cost analysis, the expert is given a functional description of the new system. He must then break the complete program down into small subprograms or modules. He then estimates the resources needed for each module and a cost analyst costs the resources predicted by the expert. The result is a dollar cost estimate for the project.

One of the problems here is that it may be difficult to define an expert and, if defined, it may be difficult to find such a person or persons. This method also requires that the functional design of the system be complete prior to the start of the estimating process. Many times the requirements of the system are not stated explicitely enough this early in the development process.

It can be argued that such detailed knowledge is essential to any type of cost estimating procedure and the requirement of a detailed, explicit breakdown of the system into small modules is really a strength of this technique rather than a weakness. Another advantage to this method is that the expert can consider interfaces and timing requirements which may be unique to this system rather than base his estimate solely on historical cost information.

The Delphi method involves a group of experts arriving at a consensus estimate. This procedure was developed by the Rand Corporation for use in other situations. A group of experts first make cost estimates individually. These estimates are then fed back to the experts and the experts make new estimates. The process iterates until a consensus is determined.

It is important to this method that the group of experts is not a committee. That is, they never meet face to face. When each estimate is made, the expert must justify and explain it. These explanations are given to the other experts, along with the estimates, all of which is anonomous.

Farquhar (11) reports on an experiment conducted by the Rand Corporation. This experiment involved the use of the Delphi method with two separate groups of experts on the same software project. Their estimates were 217 and 1090 man-months respectively. Two additional groups of experts operated as committees and arrived at estimates of 485 and 656 man-months, respectively. The actual effort expended on the project was 489 man-months, indicating that, on the basis of this experiment, the committee method is superior to the Delphi method. A later study by the Rand Corporation indicated that the Delphi method gave no better results when using experts than when using laymen. This investigation suggested that the final consensus result is often a manipulated result rather than a true consensus. All of this indicates that, when using the expert opinions technique with a group of experts, it is better to let the experts meet face to face.

PARAMETRIC EQUATION.

The use of parametric equations indicates a lot of promise for the future of software cost estimating. It is anticipated that these techniques, when they have been validated in such a way as to give estimators confidence in them, will supplant the previously mentioned techniques. The basic concept is the use of one or more non-cost parameter of the proposed system to estimate development cost.

These procedures are more sophisticated than any of those discussed previously in this report. Thus they may be viewed with suspicion by

many practitioners. Their strength however lies in the fact that they are more quantitative in their approach to the problem.

The input parameters generally include such characteristics as the number of source instructions, the type of system, the age or type of computer to be used, the type of compiler, the timing of the work, etc. Nelson (25) in his work for Systems Development Corporation, identified 94 such parameters which influence system development and therefore could be used. Of course, a procedure requiring such a great number of input factors would be useless. Also many of them are correlated and thus do not add any information to the model.

In order to be useful such a technique should require as few input parameters as possible, preferably two or three. However, when such a small number are used, other important factors will be ignored and therefore the reliability of the estimate will be lessoned. Theoretically, if a model could be developed using all 94 (or whatever portion are uncorrelated) of Nelson's factors, a highly accurate prediciton could be made. If such a model could be developed, however, no one would be able to use it computationally even if they could obtain information about all the parameters. Developers of such models must weigh accuracy of estimates against computational ease and obtain the best possible compromise.

In order to accomplish this compromise researchers interested in developing estimating models have attempted to gather as much data as possible so that the relationship between the non-cost parameters and cost can be determined. This has been the source of the first problem such researchers have faced because of the scarcity of such data which is valid. A researcher may have, for example, data on five JOVIAL, five FORTRAN and five COBOL systems. He then must decide whether to construct

three models based on five data points each or one based on fifteen. It is very tempting to take the later course because of the (erroneously assumed) increase in statistical confidence. Researchers have also pooled together data from different computers, compilers, and other factors in order to build up the sample size. This pooling of data becomes particularly important when, at the same time, a model is developed which uses a very small number of the pertinent input parameters.

As a result of the problems of limited data, data pooled in various ways, and differences in perceived predictive importance of the various input parameters, researchers have developed models which are greatly different from those developed by other researchers. The impact of this is that no model has received acceptance by persons involved in the estimating of software development costs. Since these models are so different, the next section of this report will review some of these models as they are reported in the literature. Each model will be reviewed individually for its strong and weak points.

PARAMETRIC MODELS

1. SDC Technique

This technique was developed for the Electronic Systems Division of the Air Force from 1964 to 1968. The results of the effort are contained in a nine volume report which is summarized in a handbook by E. A. Nelson (22). To gather data, the SDC researchers used questionaires with which they questioned persons who had been involved with 169 different software projects. The use of such questionaires meant that the accuracy of the data was limited to the accuracy of the responses. As mentioned previously, from these questionaires they identified 94 parameters which they attempted to relate to software cost.

They developed, by means of linear regression techniques, equations relating certain variables to cost for different development phases. Each equation uses only a partial set of the entire list of variables. These sets were selected by means of standard stepwise regression techniques.

The following table shows the distribution of data points by developer and type of program. As can be observed in the table there are many different sources of data as well as quite different types of programs included in the 169 data points. There were also quite different size programs included in the data.

Distributions of Data Sources for SDC Model

Developer		No. of Programs	Type of Program			
			Business	Scientific	Computer Software	Other
Government	U.S. Air Force	38	26	10	2	
Computer Software R & D	Company A	6	3		3	
	Company B	1			1	
	Company C	1	1		0	
	Company D	69	17	12	5	35
Computer Hardware & Aerospace	Company E	2		2		
	Company F	3	2	1		
	Company G	21	19	2		
	Company H	28	11		17	
	Totals	169	79	27	28	35

Based on all 169 points, taken together, the following equation was developed for the total cost of the design, code and test phases of software development. The use of this equation requires 14 input parameters.

$$Y_1 = -33.63 + 9.15X_3 + 10.73X_8 + 0.51X_{26} + 0.46X_{30}$$

$$+ 0.40X_{41} + 7.28X_{42} - 21.45X_{48.1} + 13.53X_{48.5}$$

$$+ 12.35X_{51} + 58.82X_{53} + 30.61X_{56} + 29.55X_{72}$$

$$+ 0.54X_{75} - 25.2X_{76}$$

where Y_1 = Total man-months required

X_3 = Lack of knowledge of operational requirements

X_8 = Stability of design

X_{26}= Percent of mathematical instructions

X_{30}= Percent of information storage and retrieval functions

X_{41}= Number of subprograms

$X_{48.1}$ = Business (for programs classified as business applications; $X_{48.1}$ = 1, otherwise $X_{48.1}$ = 0)

$X_{48.5}$ = Stand alone (for stand alone programs, $X_{48.5}$ = 1, otherwise $X_{48.5}$ = 0)

X_{51}= First program on computer (if it is a new machine to the installation and the programmers, X_{51} = 1, otherwise X_{51} = 0)

X_{53}= ADP components developed concurrently

X_{56}= Random access device used (X_{56} = 1 if used, X_{56} = 0 if not used)

X_{72}= Different computers for programming and operations

X_{75}= Number of man trips

X_{76}= Program data points developed by military organization.

It should be particularly noted that this equation does not contain as an input variable the number of computer instructions or any closely related variable. The reason for this is that there was a wide variation in effort required. For example, for 15 JOVIAL programs, they found the following results in terms of number of man-months of effort required per 1000 instructions.

Type of Instructions	Max	Min	Std. Dev.	Median	Mean
object	7.6	0.66	2.31	2.5	3.07
source	46.25	2.13	12.01	6.15	10.27

The required man-months per thousand source instructions ranged from 2.13 to 46.25 with a coefficient of variation of 117%. The fact that the mean greatly exceeds the median indicates the distribution is skewed with a few programs requiring an exceedingly large effort. All of this indicates that the mean effort would be a poor predictor for program cost estimation purposes.

They also present another version of their cost equation based on 105 of the 169 programs. These 105 programs were classified as involving large computers, i.e. computers with a purchase price of $750,000 or more.

$$Y_2 = 0.049 + 15.2X_6 - 0.23X_{25} + 0.528X_{30} + 4.50X_{37} + 0.091X_{46} - 17.5X_{48.1} + 25.1X_{51} + 22.0X_{54} + 26.0X_{56} - 0.25X_{64} - 14.9X_{65} + 10.4X_{74}$$

where

Y_2 = man-months required for large computers

X_6 = Complexity of the program system interface. (X_6 = 0, 1 or 2 depending on whether less than 10%, 10-50%, or more than 50% of the design effort is devoted to data transfer problems)

X_{25} = Percent of clerical instructions

X_{37} = Frequency of operation (X_{37} = 0, 1, 2, 3, 4, or 5 depending on the frequency of operations of the program)

X_{46} = External documentation (the number of pages written for the customers)

X_{54} = Special display equipment involving use of graphic displays, CRT's, scopes, etc. (X_{54} = 1 if used, X_{54} = 0 if not used)

X_{64} = percent of the programmers participating in program design

X_{65} = Personnel continuity, i.e. the number of personnel working for the duration of the project divided by the maximum number assigned at any time

X_{74} = Number of locations for program development

This equation uses 12 of the input variables and these twelve are somewhat different from those used in the previous equation based on all 169 data points. Specifically, X_3, X_8, X_{26}, X_{41}, X_{42}, $X_{48.5}$, X_{53}, X_{72}, X_{75} and X_{76} are used only in the first model. X_6, X_{25}, X_{37}, X_{46}, X_{54}, X_{64}, X_{65} and X_{74} are used only in the second model. X_{30}, $X_{48.1}$, X_{51}, and X_{56} are used in both. That is ten variables are used only in the first model, eight are used only in the second and only four are used in both.

The use of either of these equations requires extensive previous knowledge of a large number of software development parameters and thus lacks the requirement of parsimony. Without this knowledge the procedure cannot be used. In spite of its drawbacks, Boehm (3) identified the SDC work as the most exhaustive quantitative analysis done to date on factors influencing software development. Certainly the beginning data base is larger than any others.

Nelson (23) correctly observes that although multiple linear regression reveals important parameters, it does not necessarily reveal natural laws. Furthermore, if nonlinear regression methods had been used different parameters may have been identified. Several other studies have indicated that these relationships may not be linear as was assumed in the SCD study. In fact SDC reported on one nonlinearity. Larger projects require disproportionately more resources than smaller ones. They did not make use of this observed fact and instead got around it by eliminating or transforming data points representing large systems. Pietrasanta (28) explains

that this nonlinearity is because a linear increase in the number of programs is accompanied by an exponential increase in the potential interfaces between the programs.

2. Aron's Technique

Aron (1) published a technique for software cost estimating based on a large number and variety of major IBM system programs. He identifies the parameters of his model and recommends values for them based on his data. He does not specifically identify the data but merely indicates it contains a large number of systems.

His technique begins with a system design and an estimate of the number of deliverable assembly level instructions. The difficulty of these instructions must then be judged easy, medium or hard, based on the number of interactions contained in the program. The estimator then utilizes the factors listed in the following table to obtain the productivity to be expected in terms of assembly level instructions per man-month. After applying the appropriate productivity factors to each program, the result is multiplied by 2.5 to account for management and support personnel.

Aron's Productivity Table

Duration / Difficulty	6-12 months	12-24 months	More than 24 months	Remarks
Easy	20	500	10,000	Very Few Interactions
Medium	10	250	5,000	Some interactions
Hard	5	125	1,500	Many interactions
Units	Instructions per man-day	Instructions per man-month	Instructions Per man-year	

Aron's technique is heavily dependent on the number of instructions. It also deals solely with assembly level instructions and not high order languages. The table of productivity factors indicates a large range of productivity based on a subjective evaluation of the degree of difficulty. Aron also states that if the program extends over a period of two years, productivity increases due to a learning effect. He offers no proof of this and no other literature supports it. This learning effect can be observed in the table by observing that productivity for an easy program goes from 20 instructions per day for a 6-12 month program to 33 instructions per day (10,000/year) for a program lasting longer than two years.

3. Wolverton's Technique

Wolverton published an article concerning a cost estimating technique developed at TRW in 1974 (38). The article fails to mention the data base used except that it is TRW propietary data. It does describe the technique in some detail. The article fails to indicate how well the estimating technique fits the data base.

His procedure begins by identifying and then categorizing the software modules. The categories are old and new; easy, medium and difficult (yielding six categories). The modules are then categorized according to the type of function being performed. Six functions are listed. The final list thus contains 36 different categories.

Wolverton then provides a table of cost per instruction factors for each of these 36 categories. The number of object instructions is then multiplied by the appropriate cost factor to yield a total cost including overhead. These cost factors range from $15 to $75 and are extracted from actual historical costs incurred by TRW in 1972 dollars (14). Thus must be adjusted to put them in current dollars.

The primary strength of this method is its simplicity. It involves subjective judgements in order to categorize each program module but these are not difficult to do. The major difficulties are that it requires a valid estimate of the number of instructions and assumes a linear relationship between cost and number of instructions.

Wolverton claims this procedure has been used within TRW with good results. James (17) indicates this to be the most widely used and accepted work on software cost estimation. The methodology is applicable to large scale software systems which utilize a "structured programming" design approach. The use of structured programming insures the program will be in modular form.

4. Modified Wolverton Model

The System Evaluation Group of the Air Force Avionics Laboratory at Wright-Patterson AFB has developed a computerized version of the Wolverton model which they call the Modified Wolverton Model. This model requires as input the number of instructions for each of the six types of modules listed by Wolverton. It then utilizes ten equations to obtain the cost per instruction for each category. These equations were obtained by regression analysis using data displayed in the Wolverton Article (38).

The Modified Wolverton computer program generates costs for new and old code programs ranging in "percent difficulty" from 10-90 percent. The user subjectively selects the appropriate cost from the data spectrum generated. A brief discussion of this model and a complete listing of the computer program with sample output is in reference (17).

The following definitions of easy, medium and hard programming, developed by IBM, are an effort to somewhat quantify these characterizations which are essential for use of either Wolverton model. The percentages

are for use in the modified Wolverton model.

1. Easy (10,20, 30%). Very few interactions with other system elements. Most programs whose main function is to solve mathematical or logical problems are in this class. Easy programs generally interact with only input/output programs, data management programs or monitor programs.

2. Medium (40, 50, 60%). Some interactions with other elements. Included here are most utilities, language compilers, schedulers, input/output and data management programs. Medium programs interact with hardware functions, problem programs, monitors and other medium programs.

3. Hard (70, 80, 90%). Many interactions with system elements. All monitors and operating systems along with some special purpose programs such as conversational message processors are in this class. Any program which interacts with all or most other elements is in this class.

5. Automatic Data Processing Resource Estimating Procedures (ADPREP) Model

This technique was developed for the U.S. Army Computer Systems Command by the Planning Research Corporation (PRC) (8). The model was based on data collected from 20 Army data systems. Half of these were new systems. The remainder were major revisions of existing systems. All of the systems were business type systems and required a level of effort averaging 105 man-months. PRC provided many estimating guidelines so that the estimator can use the method of analogy to portions of the 20 systems in the data base to arrive at appropriate values for entry into a worksheet.

The technique contains a set of equations developed by means of linear regressions using the data from the twenty Army systems. The following equation, for example, predicts the total man-months of

effort to design, code and test the software.

$$Y_2 = 2.57X_2 + 5.10X_3 + 0.12X_5$$

where Y_2 = Personnel requirements in man-months

X_2 = Total number of different output formats in ADPS products

X_3 = Total number of record types in the data base

X_5 = Average number of transactions per month of input in thousands.

If this is compared to the SDC equation, it will be observed that now only three input variables are required as opposed to the 14 required previously. This model also does not rely on an estimate of the number of instructions to be written.

The ADPREP manual claims the above estimating equation has a multiple coefficient of determination, R^2, of 1.0. This seems too high to be realistic and leads one to suspect the entire model. This claim is that the equation explains all of the variance found in the sample data and that all of the data points (10 in this case) lie exactly on the hyperplane determined by the equation. If they were truly random points this does not seem likely.

The technique is very well documented in the manual but the equations should not be used outside the range of the sample data (relatively small business applications). The model assumes a lot of detailed knowledge is known about the system by the estimator. This may be feasible for a business system but is not feasible for a weapon system.

6. ESD Model

The summary notes of a workshop on software sponsored by the Electronic Systems Division of the U.S. Air Force in October, 1974 forms the basis for this model (15). The members of the workshop identified 10 factors influencing software development cost along with the relationship of each to cost. One of these factors is the number of delivered source instructions and determination of this factor is the primary step in using the model. The members of the workshop determined that the use of source instructions was a better estimating factor than object instructions, which was used in Wolverton's model.

When the number of instructions and the language are known, cost factors included in the report are used to arrive at a basic cost estimate. This estimate is then modified by the other factors. Many of these other factors are subjective along with their relation to cost. Such factors are the size and structure of the data file, the complexity of the programs and quality of personnel. Some of these could be handled quantitatively but are not in this model.

While there is something to be said for not using a fixed factor for each type of program, development technique, etc., the lack of objective cost factors for these might lead to wide variations in cost estimates by different estimators. Thus the validity of this model would be very difficult to evaluate.

7. Tecolote Model

This technique was developed by Tecolote Research, Inc. of Santa Barbara, California, for the Office of Naval Research. It is documented in a report by Brad C. Frederic published in 1975 (13). For this

report, the researchers gathered a data base of 387 points. It included the 169 points in the SDC study as well as data from TRW, North American Autonetics and 12 other sources. After gathering all these data, the researchers found they could not interpret much of it because no knowledgeable person was available to interpret the data elements. This seems to be a common problem as basic terms such as cost and instructions are not uniquely defined.

The researchers then abandoned all but five of the 387 data points. These five were judged to be the only ones about which they had sufficient information. It is interesting to note that the 169 points in the SDC study were among those abandoned.

They then performed simple regression analyses between five response variables, such as the number of man-months of effort, and a number of different independent variables. They proposed their provisional technique as an engineering scaling law rather than a set of statistical equations. The most notable aspect of the Tecolote research was that they related their dependent variables to both weapon system parameters, such as the number of targets a system must track, and software parameters, such as the number of instructions.

The reliability of the data is of such questionable nature that the model could not be used for a reliable cost estimate but it can be useful to a software manager who wishes to know how cost might vary with each of these parameters.

There are five basic cost estimating equations reported by Frederic. Each equation requires the input of one of several input variables. The user uses the input variable in which he has the most confidence. For example, if he knows the number of delivered instructions (D), the

total development cost is $0.01(D)^{1.18}$. If he wishes to use the number of operating instructions (O), the total development cost is $0.01(O)^{1.24}$. The output of this model is in 1973 dollars. None of the relationships expressed in the equations are linear ones. This is particularly noteworthy since they are developed with only five data points. It is also to be recalled that most other researchers used linear relationships.

8. IBM Model

A model developed by IBM is documented by John C. Malone in a 1975 report (20). The development of the model was based on software cost data from projects performed by IBM which employed top-down structured programming techniques and utilized the Chief Programmer Team Operations Concept. While structured programming tends to improve software reliability and maintainability it may not be efficient in terms of computer resource usage. The chief programmer approach is a good one but it may be difficult to implement because of the requirement that each programmer is assigned program modules matching his skill.

This model addresses only the software development phase. The data include costs for the development phase of both real-time and support software. IBM maintains the equations are proprietary and has not published them.

9. Naval Air Development Center Model

The Naval Air Development Center published a report based on a study conducted by them in 1971 (6). This cost estimating technique consists of several equations, which could be used for predicting total costs for research, development, test and evaluation, and production of future avionic computer systems.

There are ten basic cost modules and a computer program is listed in reference (6) to evaluate each of them. The basic software cost module (module five) is entitled programming costs. It provides an estimate of the total number of man-months required to develop a software package for an avionic system. The equation is:

$$Y = 2.8X_2 + 1.3X_3 + 33X_4 - 17X_5 + 10X_6 + X_7 - 188$$

where Y = number of man-months

X_2 = number of machine language instructions (thousands)

X_3 = number of man-miles traveled by the contractor

X_4 = number of document types produced

X_5 = average programmer's experience with the system

X_6 = number of independent consoles

X_7 = percentage of new instructions

A different module (module two) is used to compute X_2, the number of machine language instructions in the program. Thus, although there are only six independent variables in this equation, the use of module two to determine X_2 requires several more.

10. Aerospace Model

This model is taken from a 1975 report published by the Aerospace Corporation. The actual report is not available but information listed here was taken from reference (17). The cost equations are for real-time and operational support software systems. The data base for real-time systems consisted of 13 such large scale programs which were primarily airborne and space oriented. The cost equation is derived by regression analysis and is

$$\text{man-months} = 0.057\ (\text{Instructions})^{0.94}$$

The data base for operational support systems consisted of seven such systems which were both airborne and ground support systems. The resulting equation is

$$\text{man-months} = 2.012\ (\text{Instructions})^{0.404}$$

Once the number of man-months is estimated, a dollar value per man-month is used to derive the total cost. Aerospace Corporation used a figure of $5000 per man-month but this would vary for different companies. There is no indication of the validity of this model. It should be noted that the equations are nonlinear and require an estimate of the number of instructions.

11. GRC Model

M.A. Taback and M.C. Ditmore of General Research Corporation report on a model developed by GRC (37). The purpose of this model was to estimate computer software development costs from overall systems requirements. The report presents a model for software costs based on program size, computer language, complexity and hardware constraints.

As for several other models, the first step in the use of the GRC procedure is to estimate the number of instructions. This is a critical step in the procedure. One of the basic equations is

$$C_g = 0.232\ N_i^{1.43}$$

where C_g = total software development cost

N_i = number of machine language instructions

The hardware constraint modifies this equation to

$$(C_g)C = C_g \quad \frac{0.7}{1 - \sqrt{P - 0.5}} \quad \text{for } P > 0.5$$

where P = fraction of maximum speed and memory capacity utilized

$(C_g)C$ = total constrained software cost

12. RCA Price S Model

The PRICE S model is a software development cost and schedule estimation model developed by RCA PRICE Systems, Inc (32). It is based on a similar model called the PRICE model developed by RCA for estimating hardware costs. This basic hardware estimating model has apparently been quite successful.

PRICE S inputs may be listed in three categories, hard system parameters, soft system parameters and environmental parameters. Hard system input parameters are things which can be physically measured. These include the total number of executable machine instructions and the proportions of the total instructions which represent each of seven different application types.

Soft input parameters are those which cannot be measured directly. Three such parameters used in this model are software design complexity, engineering complexity, and the required system reliability. Design complexity is a measure of the inherent difficulty of the development task. It relates difficulty of the task to the shop which is to accomplish the task. Engineering complexity is a measure of how long the project should take. All of these soft parameters are not well defined and values must be assigned subjectively to them.

Environmental parameters address such factors as inflation and the rate of technological improvement. They enable the estimator to vary the economic assumptions used to compare historical data to the present.

The output of this model includes cost and schedule estimates. The cost estimates are given in the form of a 5x3 matrix. The columns of the matrix are three phases of software development: design, implementation, and test and integration. The rows of the matrix are five activities: systems engineering, programming, configuration management, documentation, and program management. The schedule outputs consist of beginning and ending times (in months) for each of the three development phases. It is interesting that the program can be run backwards with historical costs and schedule information as input. The purpose of doing this is to get estimates of software design complexity and engineering complexity values for completed programs.

Each component of a large system may be estimated individually. The total system cost is then estimated by using the separate component estimates as input to a combined system estimate. The PRICE S model accounts for additional system level work needed to implement the separate components as part of a coherent system.

The model has been implemented in a commercial time-share environment. Users must contract with RCA PRICE Systems, Inc. to use the program with a standard computer terminal.

Schneider, in a thesis prepared for the Air Force Institute of Technology, attempted to calibrate the PRICE S model (34). His data included ten software systems involved in flight or mission performance of aircraft in real-time. Thus he cautions that his conclusions do not necessarily apply to other types of software systems. He assumed, for the purpose of his research, that cost and schedule information reported by contractors is a reasonable estimate of the true cost of

the item provided. He also assumed the PRICE S model to be valid.

Schneider collected his data by the interview method. Based on these interviews he eliminated all systems except those which were operational flight systems written in assembly language. This last requirement was necessary because, at the time of the research, PRICE S was not applicable to high order languages. In addition he required that development costs had to be easily derivable from the official reports provided by the contractor to the SPO. This meant that the software development had to either be a separate line item in the contract or the contract had to be just for software development. Only two of his original ten systems met all these criteria. Therefore his conclusions are based on these two systems. He determined it was impractical to look at every module of every system. Therefore he elected to use a sequential sampling technique to choose modules for conclusion in his study. As a result of this he chose 211 of 307 modules in one system and 173 of 295 in the other. These sample sizes appeared to be adequate to properly evaluate the system cost elements.

The conclusion of Schneider's study regarding the PRICE S model were:

1. The necessary data collection procedure to calibrate the model was extremely cumbersome. Two man-weeks were required to gather data on a single software system of approximately 300 modules. This would make it impractical for very large systems.

2. Historical data is not imcompatible with the PRICE S model for systems of the size and type involved in this research.

3. Most PRICE S input parameters may be measured objectively. The principle exceptions to this are the two complexity parameters.

These would need to be related to some objectively measured parameters through an empirical study.

4. The PRICE S model is flexible enough that it may be useful in a wide range of situations. However this same flexibility indicates that it should not be used independently of other estimating techniques. It also indicates that the assumptions reflected in the input parameters should be continually questioned if the cost estimates are to be relied upon.

It is not known by this author if further development by RCA has made this model useful for high order languages. As of 1977 it was not. If it is not, this would seem to be a serious drawback to use of the model. Another drawback is the use of number of instructions per module as an input parameter. This shortcoming has been discussed elsewhere in this report. Schneider, in his research assumed the model to be valid. Evidence of such validity is needed. Even with this assumption, he found some shortcomings in the model.

13. Doty Model

This model was developed by Doty Associates, Inc. as a result of a project performed for the Rome Air Development Center. The study used nonlinear regression procedures to develop a software cost estimating model and is reported on in a final technical report written by Herd, et. al. (16). The study has a fairly large data base (129 cases after deletion of 40 systems as extraneous). Most other nonlinear studies using regression methods have used linear or log-linear procedures to relate system size to cost. There is no evidence to support a linear model. In order to use a linear model the error term in the model should be normally distributed. In order

to use a log-linear model the logarithm of the error term should be normally distributed. Neither assumption appears to be valid in the case of software cost models. Therefore the nonlinear regression procedures used by Doty appear to have more promise of validity.

The data were categorized into four types of systems: command and control, scientific, business, and utility. They then used nonlinear regression techniques to solve for the model parameters. The following table summarizes these results. All models are of the form $MM=aI^b$, where MM is the total man-months for system development, I = total instructions in the system, and a and b are model parameters calculated from the data.

System Type	Type of Instructions	a	b	R^2	Standard Error
Command and Control	Object	4.57	1.29	.78	41.1
	Source	4.09	1.26	.80	41.1
Scientific	Object	4.50	1.07	.74	72.1
	Source	7.05	1.02	.78	72.1
Business	Object	2.90	.78	.48	12.4
	Source	4.50	.78	.61	10.7
Utility	Object	12.04	.72	.30	58.1
	Source	10.08	.81	.45	51.7
All	Object	4.79	.99	-	62.2
	Source	5.26	1.05	-	50.7

While these results are quite interesting they are still inconclusive for several reasons. First, the size of the data base is still quite small considering the dispersion found in the data. The sample sizes used for the eight different models varied from a low of 21 to a high of 58. The dispersion was quite large as evidenced by the standard errors in the above table.

A second shortcoming of this study is that the largest system included was under 60,000 source lines. Most of the data represented much smaller programs (26% were from programs with less than 10,000 source instructions).

This model is really a set of mean value algorithms in which size is the only parameter even though the study identified over 40 factors with an impact on cost. As the above table indicates, these models do not explain all of the cost. In many cases the value of R^2 indicates less than 50% is explained.

The model goes from the estimate of man-months in the above table to development cost using the following equation.

$$C = (1 + k) [(MM) \times r]$$

where

C = cost of system development

r = average labor rate in dollars per MM

k = fraction of total labor costs considered to approximate either costs.

There is an algorithm included to get program size, I, as a function of the size of the data base, the number of classes, processor times, size of memory, core size, and number of message output types. However they say it is not a good predictor. There would most likely be considerable problems with classifying some programs into the four categories of this model. Many programs will have aspects of more than one category. In view of the considerable differences among the values of a and b for the models this is a significant drawback to the use of the procedure.

14. Putnam Model

This model was developed by Lawrence A. Putnam (29), (30), (31)

for the Army Computer Systems Command. It was based on earlier work by Norden (27) which was directed towards developing a model for forecasting life cycle costs for any research and development effort. The model has several things in its favor, first it requires the use of a small number of parameters. Second, it does not require the number of instructions to be estimated in advance. The input parameters it requires are a knowledge of the total system development cost and the time at which maximum effort will be expended. The total development cost may be computed from a knowledge of the maximum effort in man-years to be expended. Thus, the entire distribution of effort may be determined from knowledge of the ordinate and abscissa of the maximum point on the effort versus time curve. This input requirement avoids the controversial requirement of knowing the productivity of programmers.

There are still requirements for the somewhat subjective evaluation of "difficulty". This is defined by Putnam as the ratio of the total effort to the square of the time at which maximum effort is required. Putnam states that this ratio is linear with number of files, number of reports and number of application subprograms individually or jointly. If this is true it is certainly feasible to estimate all the input parameters for this model.

The model makes use of the Rayleigh distribution

$$y = 2K\ at\ \exp\left[-\frac{t^2}{2t_d^2} \right]$$

where K = total development effort needed

t_d = time at which the maximum effort will be expended.

$$a = \frac{1}{2t_d^2}$$

y = percent of total effort expended at time t

t = time in months of effort

The model, if correct, will give the user the percent of the total manpower required at any time interval (t, t+Δt). The model is a modification of the familiar Weibull distribution used in reliability analysis.

$$y = \frac{\beta}{\alpha}\left(\frac{t-\gamma}{\alpha}\right)^{\beta-1} \quad \exp\left[-\left(\frac{t-\gamma}{\alpha}\right)^{\beta}\right]$$

For the Weibull model K = 1, making it a probability distribution (K is the area under the curve in Putnam's version of the Rayleigh distribution). Also the transformation from the Weibull to Putnam's model is that $\alpha = \sqrt{2}\ t_d$, $\beta=2$ and $\gamma=0$. In the Weibull probability distribution, β is the shape parameter. A value of 2 for β means that the distribution has a linearly increasing rate of failures or hazard rate. The hazard rate is the instantaneous failure rate and for the Weibull distribution is given by,

$$h(t) = \frac{\beta}{\alpha}\left(\frac{t-\gamma}{\alpha}\right)^{\beta-1}$$

For Putnam's Rayleigh model this is

$$h(t) = \frac{2}{\sqrt{2}\ td}\left(\frac{t}{\sqrt{2}\ td}\right) = 2\ a\ t$$

Thus the instantaneous effort rate would be increasing linearly with time.

This model differs from all of the previously discussed models in several ways. First, it seems to have a different purpose. Its purpose is to forecast the effort (or cost) of the software system

development over its entire life cycle. That is, it can be used to predict the cost for each segment of the development period. Most other models have as their purpose the estimation of the total cost for system development but not the detailed month by month forecast.

Another difference is that some idea of the total effort needed or, at least, an estimate of the peak effort and the time at which that effort is to be expended, is needed. While these things may often be difficult to estimate in advance, they may be easier than the number of instructions needed for many of the other models. Estimates of the number of instructions are notoriously unreliable and so this model may present a viable alternative.

This model does provide a good means of adjusting the parameter estimates if early work on the system indicates it. Thus development effort needed on later portions of the project can be forecasted more accurately. This is possible to do on many other models but on the others the procedure is not so obvious. The procedure is found in a paper by Box and Palleson (4).

Putnam presents one system as an example of the use of this model. While this application is interesting and certainly applicable it does not validate the model. Additional systems must be used to determine its validity. Work should also be performed to determine the sensitivity of the model to errors in parameter estimates. Putnam states that, "Given a proper set of project parameters, K, t_d, K/t_d^2, a system can be designed to cost with only a small uncertainty". (31, p.95). If this is true, the next question is how much uncertainty can we have in our parameter estimates to obtain "a proper set" of input parameters.

RECOMMENDATIONS FOR FURTHER RESEARCH

1. This report has reviewed fourteen specific parametric estimating models as well as other methods of obtaining cost estimates for software systems. None of these models has been shown in the literature to be valid for all software system development forecasts. Some have worked well for some systems but more has not been indicated in the literature. Many of the problems involving the application of some models to systems other than those for which they were developed concern basic definitions and nomenclature.

For this reason, the first recommendation of this report is that work be done to standardize definitions of terms used by software engineers. Hardware engineers long ago have standardized their terms and thus cost estimating procedures are more reliable for hardware than for software.

This might be done through committee effort such as is used by the American National Standards Institute to obtain consensus standards. Until this standardization is accomplished it will be extremely difficult to compare estimates using the various models.

2. In spite of the above mentioned problems with definitions and nomenclature, some research should now be performed to compare the forecasting ability of the different models. Care must be exercised in such research that all terms are defined the same and all costs are computed the same in order for valid comparisons to be made. Such comparisons should be made using systems for which the cost is known.

3. Work should be conducted to determine the sensitivity of each of these models to errors in estimating input parameters. Perhaps some of this work has been done by the model builders but evidence of it is

lacking in the documentation of the models. Such an analysis is necessary to properly evaluate the different models. Even if an input parameter is hard to estimate, if it is fairly robust to estimation errors, it still may be a feasible parameter. This sensitivity analysis should determine exactly the precision needed when estimating each parameter. For example, one model may require an estimate of the number of source instructions to be within ten percent while another may only need the estimate to be within 25% to obtain the same accuracy. Thus the use of some input parameters may be feasible for some models but not others.

4. A review of the government agencies and companies who perform software development projects should be made. This is for the purpose of obtaining any available data that would be useful for the above studies. Previous review, such as those made by Systems Development Corporation discussed as model 1 of this report and by Tecolote Research, Inc. discussed as model 7 should be made use of if possible.

5. The models reported on in this report were all oriented towards cost estimating for large software systems. Perhaps, with the future use of small mini-computers and special purpose computers which are component parts of weapon systems, large systems of this type will no longer be used. A study should be made to see how the newer type computers and their accompanying software will fit the models presented here. If none of them satisfy the problem of cost estimating for these newer, smaller systems, new approaches should be made.

The Putnam model seems to be a good candidate for application to these newer computer systems. However, this must be verified with

additional research.

6. Early in this report it was mentioned that research is needed which will reveal the underlying nature of the software development process. Such work has been started by Belady and Lehman (2) and by Riordon (33).

Further work is needed to ascertain the validity of their approach to the problem. While their work has not been oriented towards the specific task of estimating software system development cost, their modelling efforts or similar efforts by others, may provide insight to the development of suitable algorithms for such purposes.

REFERENCES

1. Aron, J.D., "Estimating Resources for Large Programming Systems" in Conference on Software Engineering Techniques (Rome, Italy, October 27-31, 1969), Birmingham, England, NATO Science Committee, 1969

2. Belady, L.A. and M.M. Lehman, "A Model of Large Program Development," IBM Systems Journal, 15, No. 3, 1976, pp. 225-251.

3. Boehm, B.W., "Software and Its Impact: A Quantitative Assessment," Datamation, 19, No. 5, May, 1973, pp. 48-59.

4. Box, George E.P. and Lars Palleson, Software Budgeting Model, Mathematics Research Center, University of Wisconsin, February, 1977. (Also reproduced on pages 96-111 of reference 31).

5. Bratman, H. and Court, T., "The Software Factory," Computer, 8, No. 5, May, 1975, pp. 28-37.

6. Buck, F., et. al., A Cost-By-Function Model for Avionic Computer Systems, Vol. I, NADC-SD-7088, Naval Air Development Center, March 1971.

7. Clapp, J.A., Software Engineering - Problems and Future Developments, Tech. Rep. AD A003-422, Defense Documentation Center, Alexandria, Va., Nov., 1974.

8. Department of the Army, Management Information Systems, Handbook of ADP Resource Estimating Procedures,(ADPREP) TB 18-19-1, Headquarters, Department of the Army, Washington, D.C., 1975.

9. Devenny, Thomas J., An Exploratory Study of Software Cost Estimating at the Electronic Systems Division, Unpublished M.S. Thesis, Air Force Inst. of Technology, Tech. Rep. AD A030-162/2ST, Wright Patterson AFB, Ohio, July, 1976.

10. Dunn, Stan, "A Software Cost Estimating Technique for Communications Systems", Transactions of the Army Operations Research Symposium, 1977, pp. 728-740.

11. Farquhar, J.A., A Preliminary Inquiry Into the Software Estimation Process, Tech. Rep. AD 712-052, The Rand Corp., Santa Monica, CA., August 1970.

12. Fara, Leonard and Burt Nanus, Factors that Affect the Cost of Computer Programming, Tech Rep. AD-447-329, Systems Development Corp., Santa Monica, CA., 1964.

13. Frederic, Brad C., A Provisional Model for Estimating Computer Program Development Costs, TECOLOTE Research, Santa Barbara, CA, 1974.

14. Gaumer, W.F., A Preliminary Cost Analysis of the Communications Processor for the F-15 Joint Tactical Information Distribution Systems, M.S. Thesis, GSM/SM/765-8, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, September 1976.

15. Government/Industry Software Workshop Summary Notes, Electronic Systems Division, Hanscom Air Force Base, MA, January 1975.

16. Herd, James H. , Postak, John N., Russell, William E, Stewart, Kenneth R., Software Cost Estimation Study, Volume I, Study Results, Tech. Rep. AD A042-264/2ST Doty Associates, Rockville, MD, June 1977.

17. James, Thomas G., Software Cost Estimating Methodology, Tech. Rep. AFAL-TR-77-66, Air Force Avionics Laboratory, Wright-Patterson Air Force Base, Ohio, August 1977.

18. LaBolle, V., Development of Equations for Estimating the Cost of Computer Program Production, Tech. Rep. AD 637-760, Defense Documentation Center, Alexandria, VA, April 1966.

19. Lecht, C.P., The Management of Computer Programming Projects, American Management Assoc., Inc., New York, 1967.

20. Malone, John L., Estimating Software Life Cycle Costs, IBM, Westlake Village, CA, 1975.

21. Nashman, A.E., "Software Development Management: The Key to Quality Software Products," EASCON 74 Record, Electronics and Aerospace Systems Convention, IEEE , New York, 1974, pp. 31-35.

22. Nelson, Edward A., Management Handbook for the Estimation of Computer Programming Costs, Tech. Rep. AD 648-750, Systems Development Corp., Santa Monica, CA, March 1967.

23. Nelson, Edward A., Some Recent Contributions to Computer Programming Management, Tech. Rep. AD 680-105, Systems Development Corp., Santa Monica, CA, April 1968.

24. Nelson, Edward A., Methods of Obtaining Estimates of Computer Programming Costs: A Taxonomy, Tech. Rep. AD 665-478, Defense Documentation Center, Alexandria, VA, August 1968.

25. Nelson, Edward A. and Thomas Fleishman, Cost Reporting for Development of Information Processing Systems, Tech. Rep. AD 657-793, Systems Development Corp., Santa Monica, CA, 1976.

26. Nelson, Richard and Alan Sukert, RADC Software Data Acquisition Program, Rome Air Development Center, Griffiss AFB, 1975.

27. Norden, Peter V., "Useful Tools for Project Management," in Management of Production, M.K. Starr (Editor), Penguin Books, Inc., Baltimore, MD, 1970, pp. 71-101.

28. Pietrasanta, A.M., "Resource Analysis of Computer Program System Development" in On the Management of Computer Programming, G.F. Weinwurm, (Ed) Auerbach, Inc., Princeton, N.J., 1970, pp. 67-87.

29. Putnam, Lawrence A., "ADP Resource Estimations: A Macro-Level Forecasting Methodology for Software Development" Proceedings, 15th Annual U.S. Army Operations Research Symposium, October 1967, pp. 323-332.

30. Putnam, Lawrence A., "The Influence of the Time-Difficulty Factor in Large Scale Software Development," Working Papers of the Software Life Cycle Management Workshop, U.S. Army Computer Systems Command, August 1977, pp. 307-312.

31. Putnam, Lawrence A., The Software Life Cycle," Quantitative Management: Software Cost Estimating Tutorial, Institute of Electrical and Electronics Engineers, New York, November 1977.

32. RCA PRICE Systems, Inc., Preliminary - May 1977, Reference Manual, PRICE Software Model, Moorestown, N.J., 1977.

33. Riordon, J.S., "An Evolution Dynamics Model of Software Systems Development," Working Papers of the Software Life Cycle Management Workshop, U.S. Army Computer Systems Command, August 1977, pp. 339-360.

34. Schneider, John, IV, A Preliminary Calibration of the RCA PRICE S Software Cost Estimation Model, M.S. Thesis, Air Force Institute of Technology, AFIT/GSM/SM/77S-15, Tech. Rep. AD A046-808/2ST, Wright Patterson AFB, Ohio, September 1977.

35. Schwartz, Jules I., "Construction of Software: Problems and Practicalities," in Practical Strategies for Developing Large Software Systems, edited by E. Horowitz, Addison-Wesley Publishing Co., Reading, MA, 1975.

36. Smith, D., "An Organization for Successful Project Management," Proc. AFIPS SJCC, 40, AFIPS Press, Montvale, N.J., 1972, pp. 129-140.

37. Taback, M.A. and M.C. Ditmore, Estimation of Computer Requirements and Software Development Costs, RM-1873, General Research Corporation, March 1974.

38. Wolverton, Ray W., "The Cost of Developing Large-Scale Software," IEEE Transactions on Computers, 23, No. 6, 1974, p. 629.